



Angular for Beginners: Part 2 Modules, Components & Directives.

By Tuba Mansoor

In this [first part](#) of this series we discussed how to get started with Angular.

In this second part we will discuss Modules, Components & Directives in Angular. It is recommended that you go through the first part, as the demo in this article is in continuation to the earlier part.

Modules

A module in angular is a group of components, directives, services etc. We will be having a look at each of these terms soon. Basically, think of module as a group.

By default we will have a file `app.module.ts` generated as below. It will be decorated with the directive `@NgModule`, which tells angular all the components, directives etc. that we will be using in our module.

App.module.ts file will list all the other modules and components that are imported. You will find that the first three lines import external modules. The Browser module is imported from the platform-browser package situated in the src folder. The second line imports the **NgModule** that is required because we are using the directive `@NgModule`. The third line imports the **AppRoutingModule**, which we will discuss when we learn about routing.

These external modules that are imported are mentioned in the imports section.

The fourth import, imports `AppComponent`. This is a component that is present in our current module. Hence it is mentioned in the declaration section.

You will find that underneath the providers, there is bootstrap mentioned. This specifies angular to load `AppComponent` at startup.

```
1 import { BrowserModule } from '@angular/platform-browser';
2 import { NgModule } from '@angular/core';
3
4 import { AppRoutingModule } from './app-routing.module';
5 import { AppComponent } from './app.component';
6
7 @NgModule({
8   declarations: [
9     AppComponent
10  ],
11  imports: [
12    BrowserModule,
13    AppRoutingModule
14  ],
15  providers: [],
16  bootstrap: [AppComponent]
17 })
18 export class AppModule { }
19
```

Where is AppComponent declared? Goto app.component.ts file and you will find it there.

```
1 import { Component } from '@angular/core';
2
3 @Component({
4   selector: 'app-root',
5   templateUrl: './app.component.html',
6   styleUrls: ['./app.component.css']
7 })
8 export class AppComponent {
9   title = 'Welcome!';
10 }
11
```

Remember how we discussed angular is a single page application? This page is index.html. If you navigate to the index.html, which is your default page, it will have <app-root></app-root>.

```
<> index.html x TS app.component.ts TS app.module.ts {} tsconfig.app.json
1 <!doctype html>
2 <html lang="en">
3 <head>
4   <meta charset="utf-8">
5   <title>Welcome!!</title>
6   <base href="/">
7
8   <meta name="viewport" content="width=device-width, initial-scale=1">
9   <link rel="icon" type="image/x-icon" href="favicon.ico">
10 </head>
11 <body>
12 <app-root></app-root>
13 </body>
14 </html>
15
```

This app-root can be found below:

```
<> index.html TS app.component.ts x TS app.module.ts {} tsconfig.app.json
1 import { Component } from '@angular/core';
2
3 @Component({
4   selector: 'app-root',
5   templateUrl: './app.component.html',
6   styleUrls: ['./app.component.css']
7 })
8 export class AppComponent {
9   title = 'Welcome!';
10 }
11
```

So basically, angular knows that it has to show AppComponent on startup.

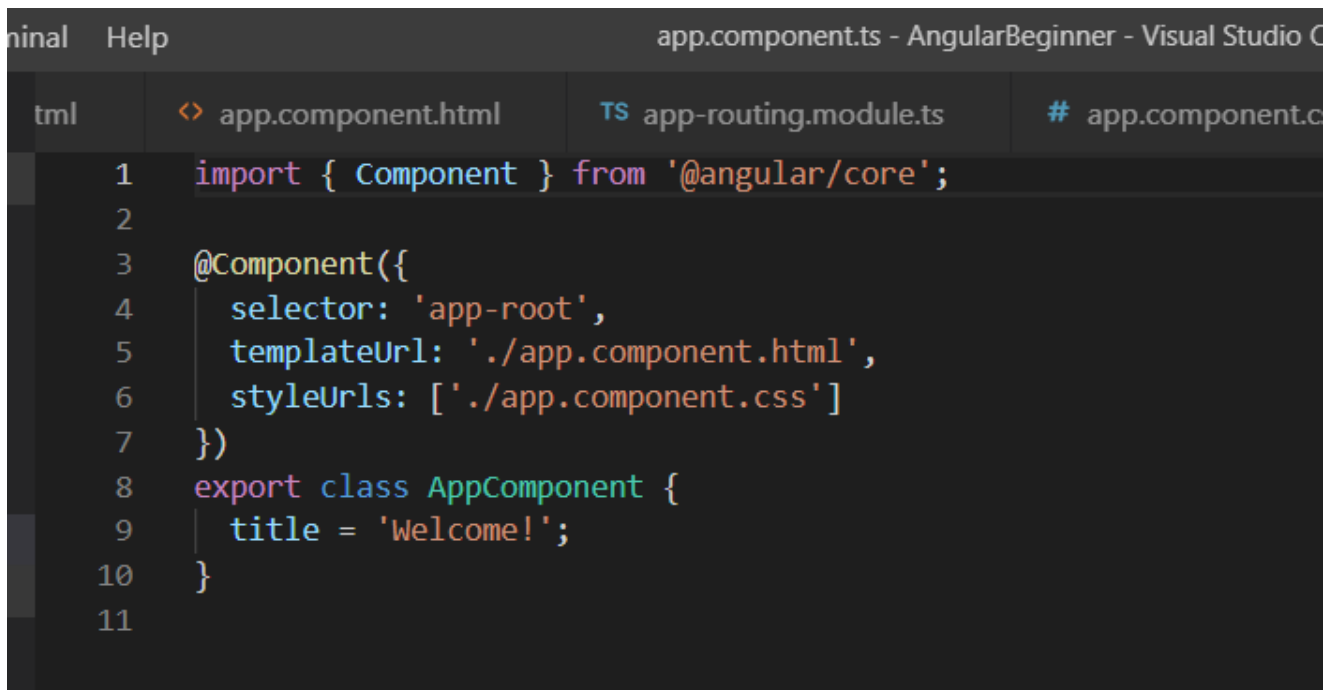
Before we move on to Components, can we create multiple modules in our application? Yes sure we can. We will always have a root module, which will be starting module and the other modules will be called feature modules. To use them, you have to import them first in you root module.

When we have a huge application, it is a good idea to have multiple modules for implementing different functionalities. A shared module can be created for housing shared functionalities,

Components

Components is the view, similar to the view that we have in MVC architectures. It will consist of template(html) + class (properties,methods written in Typescript) and metadata (some extra information).

Goto app.component.ts



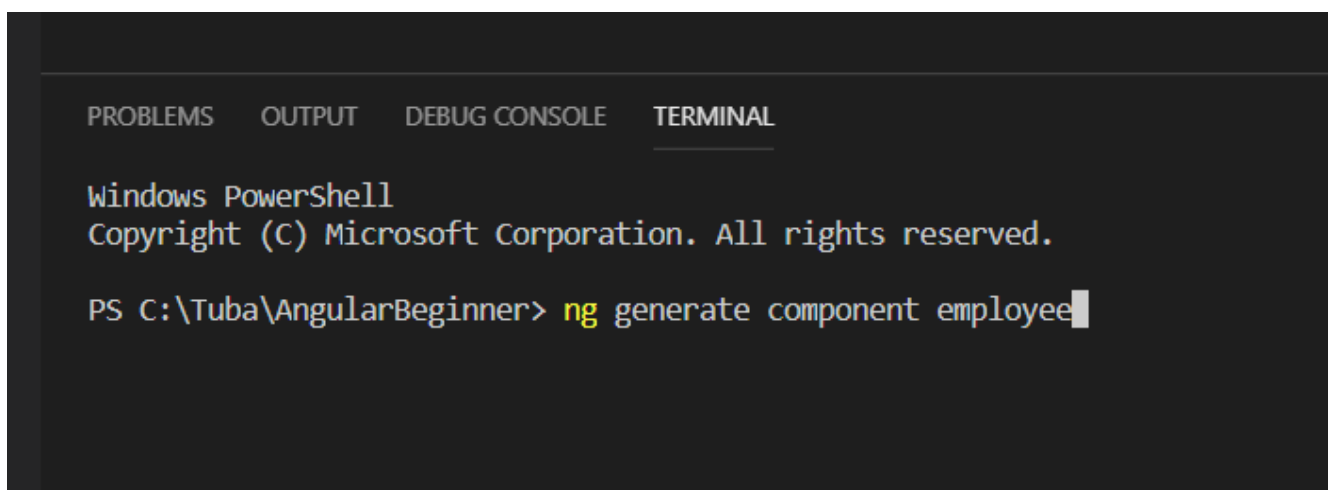
```
1 import { Component } from '@angular/core';
2
3 @Component({
4   selector: 'app-root',
5   templateUrl: './app.component.html',
6   styleUrls: ['./app.component.css']
7 })
8 export class AppComponent {
9   title = 'Welcome!';
10 }
11
```

Here you have an AppComponent class in typescript. You also have a templateUrl that specifies the html for the view and the rest is all metadata.

Let's create a component Employee.

You can create it by writing the following command in the CLI. For this create a new terminal.

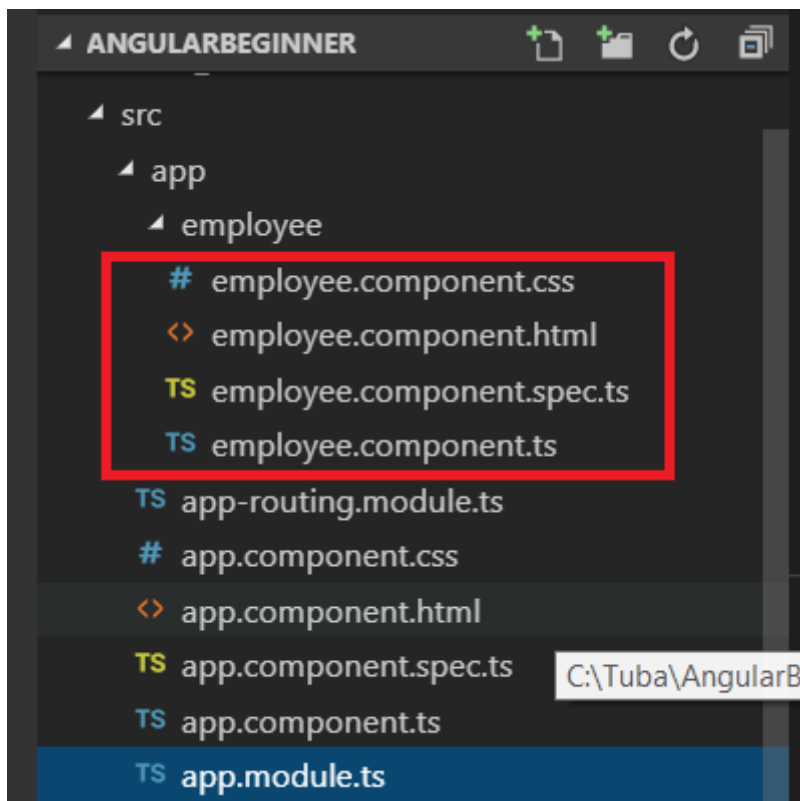
ng generate component employee



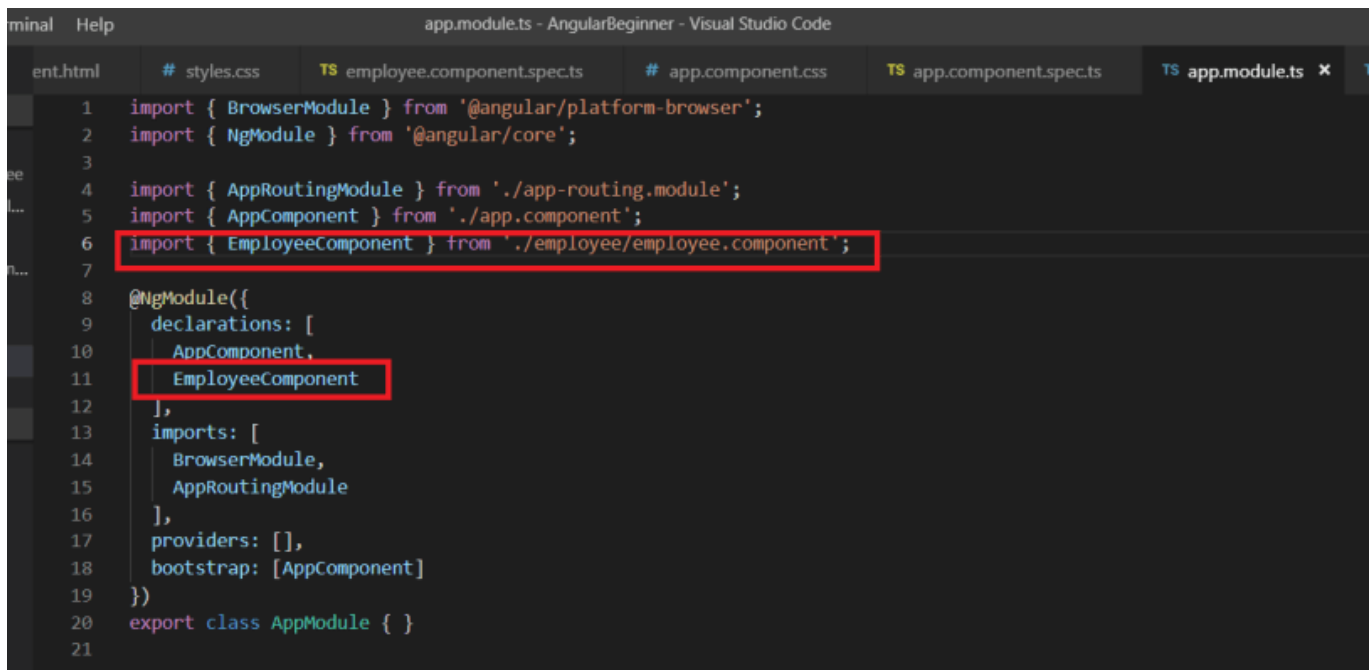
```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Tuba\AngularBeginner> ng generate component employee
```

This will create a new component employee. Angular will create four files for you as part of the component:



Also when we had created the employee component through angular CLI, angular registered our component class in the root module as below:



Directives

Directives help structure and modify your DOM. There are three types of directives in angular:

Components:

The templates in components structure your DOM. We have already seen them.

Structural Directives:

Change the layout of the DOM. For example, we have **ngIf** and **ngElse** statements that help decide whether to display a particular element or not.

Now let's create an employee class as follows. We don't know its type, so we have defined it as an array of any type. We can create an employee class separately and we could have defined employees as an array of that class. But for our demo purpose, we don't need it.

```
1     employees : any[] =[ {
2         "employeeId" : 1,
3         "employeeName" : "Tuba",
4         "projectId":100
5     },
6     {
7         "employeeId" : 2,
8         "employeeName" : "Atul",
9         "projectId":101
10    },
11    {
12        "employeeId" : 3,
13        "employeeName" : "Theran",
14        "projectId":101
15    }
16    ]
```

```
terminal Help employee.component.ts - AngularBeginner1 - Visual Studio Code
<> index.html TS app.component.ts TS employee.component.ts x TS app.module
6 styleor ts. [ './employee.component.css' ]
7 })
8 export class EmployeeComponent implements OnInit {
9
10 constructor() { }
11
12 ngOnInit() {
13 }
14
15 employees : any[] = [ {
16   "employeeId" : 1,
17   "employeeName" : "Tuba",
18   "projectId":100
19 },
20 {
21   "employeeId" : 2,
22   "employeeName" : "Atul",
23   "projectId":101
24 },
25 {
26   "employeeId" : 3,
27   "employeeName" : "Theran",
28   "projectId":101
29 }
30 ]
31
```

Before we display employees in our view, lets add bootstrap, so that we can get a responsive UI. This can be done easily by typing in the following command:

npm install bootstrap

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Tuba\AngularBeginner> npm install bootstrap
```

Before we can use bootstrap classes in our employee.component.html, we need to import the styles in style.css as below:

```
1 @import "~bootstrap/dist/css/bootstrap.min.css"
```

```
> index.html TS app.component.ts TS employee.component.ts # styles.css x ↺ □ ·
1 /* You can add global styles to this file, and also import other style
2 @import "~bootstrap/dist/css/bootstrap.min.css"
```

Write the following code in employee.component.html:

```
1 <div class="container">
2   <h2>Employee List</h2>
3   <div class="table-responsive">
4     <table class="table" *ngIf = 'employees && employees.length' >
5       <thead>
6         <tr>
7           <th>Employee Id</th>
8           <th>Employee Name</th>
9           <th>Project Id</th>
10        </tr>
11      </thead>
12      <tbody>
13        <tr *ngFor = 'let employee of employees'>
14          <td>{{employee.employeeId}}</td>
15          <td>{{employee.employeeName}}</td>
16          <td>{{employee.projectId}}</td>
17        </tr>
18      </tbody>
19    </table>
20    <div>
21  </div>
22
23
```



```
terminal Help employee.component.html - AngularBeginner - Visual Studio Code
<div class="container">
  <h2>Employee List</h2>
  <div class="table-responsive">
    <table class="table" *ngIf = 'employees && employees.length' >
      <thead>
        <tr>
          <th>Employee Id</th>
          <th>Employee Name</th>
          <th>Project Id</th>
        </tr>
      </thead>
      <tbody>
        <tr *ngFor = 'let employee of employees'>
          <td>{{employee.employeeId}}</td>
          <td>{{employee.employeeName}}</td>
          <td>{{employee.projectId}}</td>
        </tr>
      </tbody>
    </table>
  </div>
</div>
```

Here we have used two structural directives : ngIf & ngFor. If employees exists in the employee.component.ts file and it's length is more than 0, then the table will be displayed. Also we have used the ngFor structural directive to loop over the employees.

Structural directives always begin with an '*'.

We also need to add the employee component directive to app.component.html as below, so that it is displayed in the browser.

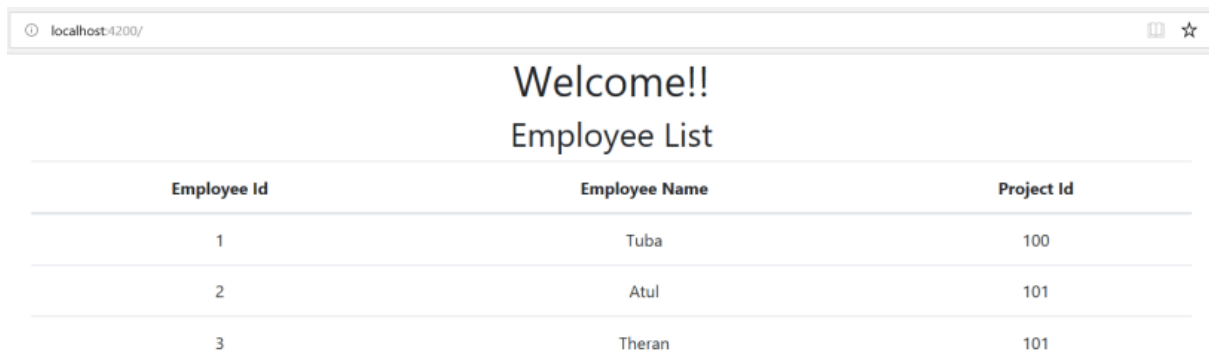
```
1 <app-employee></app-employee>
```

```
terminal Help app.component.html
<!--The content below is only a placeholder
<div style="text-align:center">
  <h1>
    | {{ title }}!
  </h1>
  <app-employee></app-employee>
  <router-outlet></router-outlet>
```

The directive 'app-employee' is mentioned in the metadata of employee.component.ts:

```
g Terminal Help employee.component.ts - AngularBeginner - Visual Studio Code
<> app.component.html TS app-routing.module.ts TS employee.component.ts x <>
1 import { Component, OnInit } from '@angular/core';
2
3 @Component({
4   selector: 'app-employee',
5   templateUrl: './employee.component.html',
6   styleUrls: ['./employee.component.css']
7 })
8 export class EmployeeComponent implements OnInit {
9
10  constructor() { }
11
12  ngOnInit() {
13
14  }
15
16  employees : any[] =[ {
17    "employeeId" : 1,
18    "employeeName" : "Tuba",
19    "projectId":100
20  },
21  {
22    "employeeId" : 2,
23    "employeeName" : "Atul",
24    "projectId":101
```

Now let's go to our browser. We can see a table displayed as below:



Attribute Directives

These are directives that change the appearance or behaviour of a DOM element. For example: `ngStyle`.

Let's modify our code to use `ngStyle` as below:

```
1 <td [ngStyle]="{'background-color':employee.projectId == '101' ? 'green' : 'red'
   }">{{employee.projectId}}</td>
```

```
TS employee.component.ts  employee.component.html  # employee.component.css  # styles.css  TS app.module.ts  app.component.html
1 <div class="container">
2 <h2>Employee List</h2>
3
4 <div class="table-responsive">
5 <table class="table" *ngIf = 'employees && employees.length' >
6 <thead>
7 <tr>
8 <th>Employee Id</th>
9 <th>Employee Name</th>
10 <th>Project Id</th>
11 </tr>
12 </thead>
13 <tbody>
14 <tr *ngFor = 'let employee of employees'>
15 <td>{{employee.employeeId}}</td>
16 <td>{{employee.employeeName}}</td>
17 <td [ngStyle]="{'background-color':employee.projectId == '101' ? 'green' : 'red' }">{{employee.projectId}}</td>
18 </tr>
19 </tbody>
20 </table>
21 </div>
22 </div>
```

The above condition will apply a particular style after evaluating a particular expression. If the projectId is 101, the background for the column will be green else red. Our browser will look as below:

Welcome!!
Employee List

Employee Id	Employee Name	Project Id
1	Tuba	100
2	Atul	101
3	Theran	101

This was all about directives. Let's proceed to understand the different types of data bindings in Angular in the next part of this Angular Beginner's course.

Happy learning!

This article was originally published on my website [taagung](http://taagung.com).